**Annotations Add in before API**

@PreAuthorize("hasAnyRole('ADMIN','DB-ADMIN')")

@PreAuthorize("hasRole('ROLE\_USER')")

**Impl File for Generate Token**

package com.maan.eway.auth.token;

import static com.maan.eway.auth.token.Constants.ACCESS\_TOKEN\_VALIDITY\_SECONDS;

import static com.maan.eway.auth.token.Constants.SIGNING\_KEY;

import java.io.Serializable;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Arrays;

import java.util.Date;

import java.util.function.Function;

import javax.servlet.http.HttpServletRequest;

import org.apache.commons.lang3.StringUtils;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Component;

import com.maan.eway.bean.LoginMaster;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

@Component

public class JwtTokenUtil implements Serializable {

private static final long serialVersionUID = 1L;

public String getUsernameFromToken(String token) {

return getClaimFromToken(token, Claims::getSubject);

}

public Date getExpirationDateFromToken(String token) {

return getClaimFromToken(token, Claims::getExpiration);

}

public <T> T getClaimFromToken(String token, Function<Claims, T> claimsResolver) {

final Claims claims = getAllClaimsFromToken(token);

return claimsResolver.apply(claims);

}

private Claims getAllClaimsFromToken(String token) {

return Jwts.parser().setSigningKey(SIGNING\_KEY).parseClaimsJws(token).getBody();

}

private Boolean isTokenExpired(String token,HttpServletRequest req) {

Date expiration = new Date();

boolean time = false;

String value = req.getSession().getAttribute(token)==null?"":req.getSession().getAttribute(token).toString();

if(StringUtils.isNotBlank(value)) {

DateFormat formatter = new SimpleDateFormat("E MMM dd HH:mm:ss Z yyyy");

try {

expiration = formatter.parse(value);

long addMinuteTime = expiration.getTime();

expiration = new Date(addMinuteTime + (30\*60\*1000));

time = expiration.before(new Date());

if(!time) {

req.getSession().setAttribute(token,new Date());

}

} catch (ParseException e) {

e.printStackTrace();

}

}else {

req.getSession().setAttribute(token,new Date());

}

return time;

}

public String generateToken(LoginMaster user) {

return doGenerateToken(user.getLoginId());

}

public String doGenerateToken(String loginid) {

Claims claims = Jwts.claims().setSubject(loginid);

claims.put("scopes", Arrays.asList(new SimpleGrantedAuthority("ROLE\_ADMIN")));

claims.put("scopes", Arrays.asList(new SimpleGrantedAuthority("ROLE\_USER")));

String token = Jwts.builder()

.setClaims(claims)

.setIssuer("http://devglan.com")

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + 1800000 ))

.signWith(SignatureAlgorithm.HS256, SIGNING\_KEY)

.compact();

return token;

}

public Boolean validateToken(String token, UserDetails userDetails,HttpServletRequest req) {

final String username = token;//getUsernameFromToken(token);

return (username.equalsIgnoreCase(userDetails.getUsername()) && !isTokenExpired(token,req));

}

}